COBEM-2017-2056
TOPOLOGY OPTIMIZATION FOR SEVERAL BOUNDARY CONDITIONS IN STATICALLY LOADED STRUCTURES

Lucas Soares Dornelles
Federal University of Rio Grande (FURG), School of Engineering, Rio Grande, RS.
lucas.dornelles@furg.br

Daniel Milbrath De Leon
Federal University of Rio Grande do Sul (UFRGS), Applied Mechanics Group (GMAp), Mechanical Engineering Department, Porto Alegre, RS

Abstract. *This work presents a compilation of several topology optimization problems of boundary conditions for compliance minimization in statically loaded structures. The code is written using the open source software Scilab, then anyone is allowed to reproduce the results. The main goal of this work is to provide very simple routines for topology optimization problems, giving to the engineering student a lot of insight while running the code. The tool developed intends to foster the study of topology optimization, using simple structures and boundary conditions to introduce the concept of optimization for the engineering students.*

Keywords: *topology optimization, optimality criteria, compliance minimization.*

1. INTRODUCTION

In topology optimization problems, we seek for the best material distribution (Ω_d) inside a prescribed domain (Ω), this domain is bounded by some natural and essential boundary condition (Γ), as can be seen in Fig.1. The material is often taken as a pseudo-material model, commonly named as ρ, and a function is extremized in a way that some constraints are satisfied. However, by the simple application of this concept to solve the problem, i.e with a discrete valued optimization, the problem becomes ill posed and a unique solution does not exist. The approximation commonly used to overcome this difficulty is to change integer variables by continuum ones, introducing some kind of penalization to drive the solution back to the discrete form (Bendsøe and Sigmund, 2003).

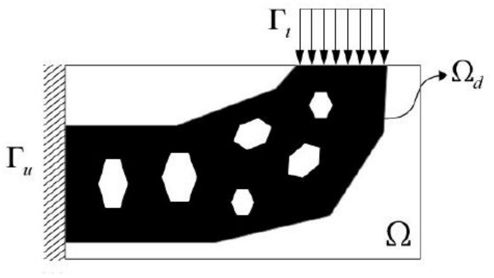

Figure 1. Example of a topology optimization problem domain.

A method that performs very well and fulfills this issue is the power law approach or solid isotropic material with penalization (SIMP) (Bendsøe, 1989). The material density distribution is scaled element wise in the finite element mesh to values ranging between 0 and 1, changing the problem to be well-posed. Then, the optimization problem can be solved by some mathematical programming method. A very simple mathematical algorithm, applied in topology optimization problems, mainly for compliance problems, is the optimality criteria (Bendsøe, 1995), where a couple of lagrangian multipliers are updated.

Nowadays, topology optimization is a very known tool in order to design structures looking for material economy and wise applications. Although very rare in engineering courses, this structural optimization branch is very important in the modern design approaches.

This work is inspired in the work of Sigmund, 2001. The main goal is to apply its developed routines to an open source software and to extend the examples suggested. The purpose of this work is fully academic, and the developed code is available on the appendix (section 6).

2. TOPOLOGY OPTIMIZATION PROBLEM

The mathematical modeling of the topology optimization follows the same simplifications as in Sigmund, 2001. The design domain is discretized by square finite elements, assuming a rectangular shape with its aspect ratio given by the ratio of elements in the horizontal and vertical direction. It is important to note that even though the design domain is restricted on a rectangular shape, with the use of passive elements is possible to alter its format as desired.

The topology optimization problem by compliance minimization based on the SIMP method can be written as follow.

$$\begin{aligned} min_x &: \; c(x) = U^T K U = \sum_{e=1}^{N} (x_e)^p u_e^T k_0 u_e \\ subject\ to &: \; \frac{V(x)}{V_0} = f \\ &: \; KU = F \\ &: \; 0 < x_{min} \le x \le 1 \end{aligned} \tag{1}$$

where K is the global stiffness matrix, U the global displacement matrix, k_0 the element stiffness matrix, u_e the element displacement vector, x the vector of design variables, p the SIMP penalty, N the number of elements used to discretize the design domain, F the force vector, $V(x)$ the material volume, V_0 the design domain volume and x_{min} is a prescribed low value imposed to avoid numerical instability.

This optimization problem is solved using the optimality criteria method, as described in section 6 Appendix, lines 90 to 111.

3. COMPUTATIONAL PROCEDURE

For a better visualization, the present Scilab code was divided into sections represented in the flowchart in Fig. 2 being the initialization section where the problem is stated. The Scilab code is run utilizing Scilab editing tool Scinotes and the densities matrix are plotted utilizing the Scilab Image and Video Processing toolbox extension (SIVP http://sivp.sourceforge.net/). The main differences when compared to Sigmund (2001) are the finite element analysis and objective function positions, this change was made in order to display the actual iteration value. Another difference is the scaling of the densities matrix for better visualization when plotted. The two dotted section seen on the flowchart represent the preparation phase and iteration phase of the algorithm.

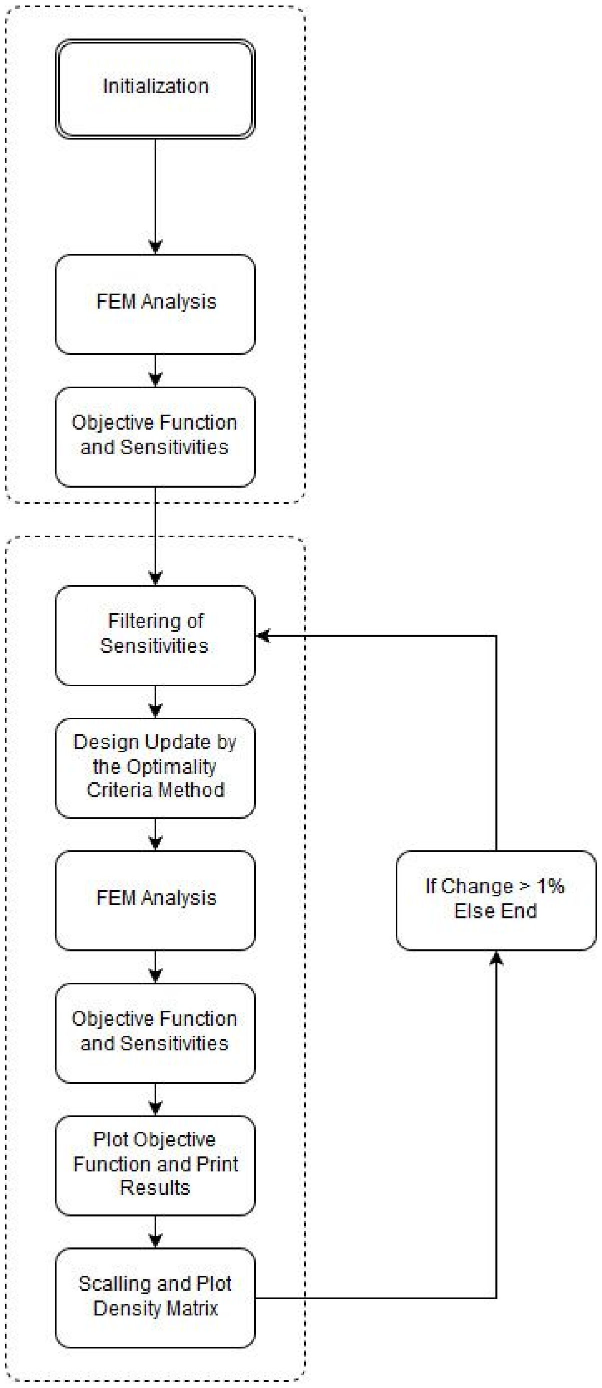
Initialization
FEM Analysis
Objective Function and Sensitivities
Filtering of Sensitivities
Design Update by the Optimality Criteria Method
FEM Analysis
Objective Function and Sensitivities
Plot Objective Function and Print Results
Scalling and Plot Density Matrix
If Change > 1% Else End

Figure 2. Computational Procedure Flowchart

Another change was made on the FEM Analysis section where the Scilab function umfpack is used to solve the set of linear equations instead of the usual backslash command (Appendix, lines 51-52 and 126-127). The reason for this is that the umfpack function has shown to be faster on the solved problems with the same reliability. The following subsection is a description of the variables found on the initialization phase of the algorithm used to the problem definition.

3.1 Initialization

The initialization section (Appendix, lines 3-22) is where the problem is defined, the variables are explained below:

-nelx; Integer, number of elements on the x axis;
-nely; Integer, number of elements on the y axis;
-volfrac; Double, volume fraction for the volume restriction;
-penal; Integer, penal power for the SIMP method, usually 3;

-rmin; double, radius of the sensitivities filter;

-scale; Integer, scaling coefficient for the densities matrix plot;

-passive; matrix, passive elements matrix, elements with value 1 in the passive matrix are passive elements;

-F; Sparse vector, load position and module;

-fixeddofs; matrix, matrix containing the value of fixed degrees of freedom's positions;

Beside these variables there are E and ν, which are the Young's moduli and poisson ration of the material, respectively.

The elements and nodes are numbered column wise and each node has two degrees of freedom (x and y axis).

4. TOPOLOGY OPTIMIZATION PROBLEMS

Here, we show the solution for the verification problem compared with that in Sigmund (2001) (Fig. 4). The code used to solve the verification problem can be found on the Appendix section (Section 6).

The problem consists of a MMB beam with a unitary force applied on its center (Fig. 3), for the test is used half the beam, with its symmetric boundary conditions, with the force applied on its upper left. The half beam is composed of 60 isoparametric finite elements on the horizontal and 20 on the vertical (respectively x and y axis), the poisson ratio used is 0.3, the penalty power for the power law approach is 3, the minimum radius for the mesh independency filter is 1.5 and the initial density of the finite elements and the volume fraction restriction is 0.5. The convergence criterion was the difference in the material distribution for two subsequent iterations be less than 1%.

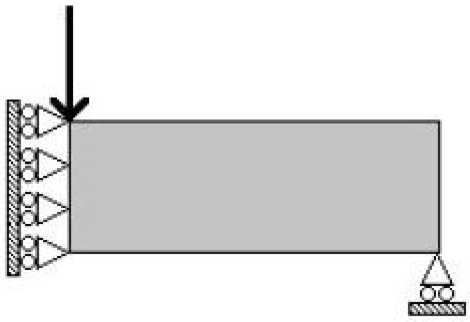
Figure 3. MMB problem

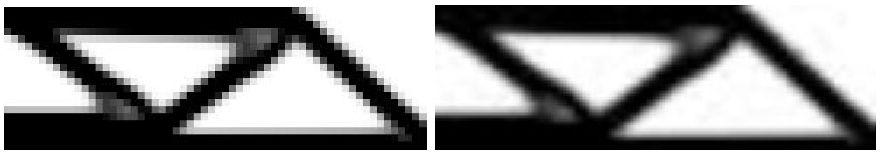
Figure 4. Left: Scilab result, right: Matlab result (Sigmund, 2001).

As it can be seen, the code written on Scilab was able to reproduce the results obtained by the matlab code, the time elapsed by iteration was approximately 1.6 seconds on a 2.93 GHz i3 CPU.

The following subsections are dedicated to the expansion of the existing examples of solution of topology optimization problems, changing the design domain and applied forces, introducing distributed loads, passive elements and non symmetric problems.

4.1 Passive elements problem

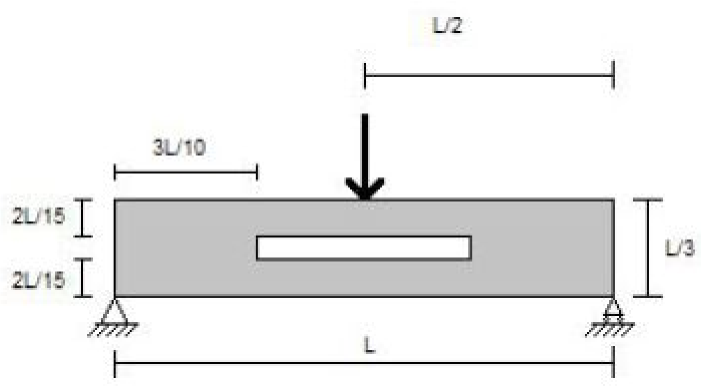

Figure 5. Passive elements problem

Since the passive elements finder and passive elements matrix were already implemented on the written code, only the initialization phase needs to be changed to solve the passive elements problem in Fig. 5. In order to run this problem, the following modifications must be performed:

```
5   nelx=90
6   nely=30
19  F(2*(nely+1)*45+2,1)=-1
20  fixeddofs=union((2*(nely+1)-1):(2*(nely+1)),2*(nely+1)*(nelx+1))
```

With the passive element matrix placed at line 14:

```
14  passive(12:nely-11,27:nelx-26)=1
```

The stated problem was solved utilizing two different volume fractions and the results are shown in Fig. 6.

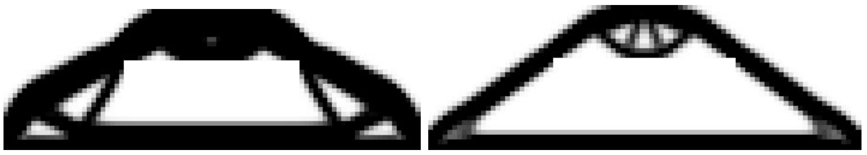

Figure 6. Passive elements problem results, left: volume fraction 0.5, right: volume fraction 0.3

4.2 Non symmetric problem

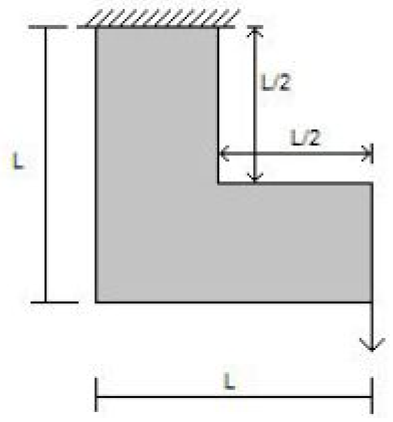

Figure 7. Non symmetric problem

To limit the design space on the problem stated in Fig. 7 it was used passive elements and the length used was 30 elements. The modified variables are as follow

```
5   nelx=30
```

```
6   nely=30
19  F((2*(nely+1)*(nelx+1)),1)=-1
20  fixeddofs=union(1:2*(nely+1):(2*(nely+1)*nelx)+1,2:2*(nely+1):(2*(nely+1)*nelx)+2)
```

The passive elements matrix placed at line 14:

```
14  passive(1:15,16:30)=1
```

The resulting designs are shown in Fig. 8.

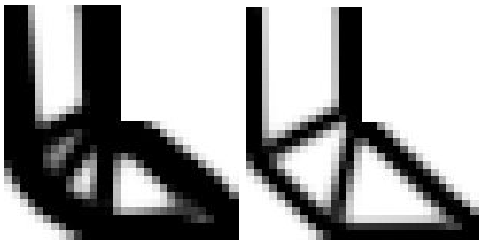
Figure 8. Non symmetric problem results, left: volume fraction 0.5, right: volume fraction 0.3

4.3 Distributed load

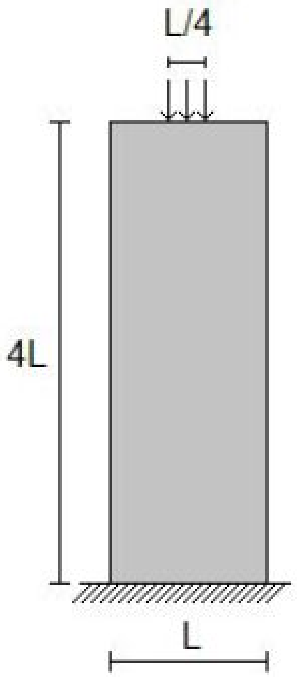

Figure 9. Distributed load problem

Using the length L=20 the number of elements for the Fig.9 problem are as follow:

```
5   nelx=20
6   nely=80
```

And the distributed load must be written as a matrix, placing the load on the affected nodes degrees of freedom:

```
19  F=sparse([],[],[2*(nely+1)*(nelx+1),5])
20  F((2*(nely+1))*8+2,1)=-1; F((2*(nely+1))*9+2,2)=-1; F((2*(nely+1))*10+2,3)=-1; F((2*(nely+1))*11+2,4)=-1;
F((2*(nely+1))*12+2,5)=-1
```

In addition to the changes made on the initialization phase variables it is necessary to modify the objective function to a sum of the compliances for each force applied. The full objective function and sensitivities sections (lines 54-64 and 129-139) are substituted for the following lines:

```
//OBJECTIVE FUNCTION AND SENSITIVITY
c=0
for ely=1:nely
   for elx=1:nelx
      n1=(nely+1)*(elx-1)+ely
      n2=(nely+1)*elx+ely
      dc(ely,elx)=0
      for i=1:5
         Ue=U([2*n1-1;2*n1;2*n2-1;2*n2;2*n2+1;2*n2+2;2*n1+1;2*n1+2],i)
         c=c+x(ely,elx)^penal*Ue'*KE*Ue
         dc(ely,elx)=dc(ely,elx)-penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue
      end
   end
end
```

The resulting designs are shown in Fig. 10.

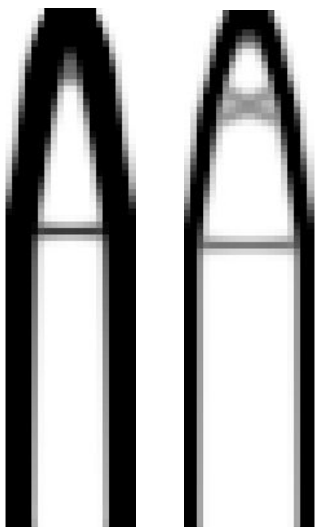
Figure 10. Distributed load problem results, left: volume fraction 0.5, right: volume fraction 0.3

5. CONCLUSION

Although very important, topology optimization problems are not a simple task to be performed by beginners. Because of the high number of parameters involved, this approach can be cumbersome on a first attempt. Every tool that provides an educational understanding of this discipline is of great value.

By means of an open source software and routines that are easy to modify, some common examples are depicted in this paper as well as comments that allow the user to modify boundary conditions and domain. The main contribution of this work is to foster the use of topology optimization to solve structural problems, giving a first insight to students that aims to make their first researches on this subject.

6. APPENDIX

```
1    //Topology Optimization on Scilab
2    //This code is based on Sigmund's 99 lines Matlab code which can be found on www.topopt.dtu.dk
3    //////INITIALIZATION//////
4    clear
5    nelx=60
6    nely=20
```

```
volfrac=0.5
penal=3.0
rmin=1.5
scale=5

x(1:nely,1:nelx)=volfrac
passive(1:nely,1:nelx)=0

loop=0
change=1

//LOADS AND SUPPORTS
F=sparse([],[],[2*(nely+1)*(nelx+1),1])
F(2,1)=-1
fixeddofs=union([1:2:2*(nely+1)],[2*(nelx+1)*(nely+1)])
alldofs=[1:2*(nely+1)*(nelx+1)]
freedofs=setdiff(alldofs,fixeddofs)

////// ELEMENT STIFFNESS MATRIX //////
E=1
nu=0.3
k=[1/2-nu/6 1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 -1/4+nu/12 -1/8-nu/8 nu/6 1/8-3*nu/8]
KE=E/(1-nu^2)*[k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
            k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
            k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
            k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
            k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
            k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
            k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
            k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)]

////// FE-ANALYSIS //////
K=sparse([],[],[2*(nelx+1)*(nely+1),2*(nelx+1)*(nely+1)])
U=sparse([],[],[2*(nely+1)*(nelx+1),1])
for ely=1:nely
   for elx=1:nelx
      n1=(nely+1)*(elx-1)+ely
      n2=(nely+1)*elx+ely
      edof=[2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2]
      K(edof,edof)=K(edof,edof)+x(ely,elx)^penal*KE
   end
end

//SOLVING
U(freedofs,:)=umfpack(K(freedofs,freedofs),"\",full(F(freedofs,:)))
U(fixeddofs,:)=0

//OBJECTIVE FUNCTION AND SENSITIVITY
c=0
for ely=1:nely
   for elx=1:nelx
      n1=(nely+1)*(elx-1)+ely
      n2=(nely+1)*elx+ely
      Ue=U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1;2*n2+2; 2*n1+1;2*n1+2], 1)
      c=c+x(ely,elx)^penal*Ue'*KE*Ue
      dc(ely,elx)=-penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue
   end
end
```

```

//START ITERATION
while change>0.01
   tic
   loop=loop+1
   xold=x

//FILTERING OF SENSITIVITIES
//////MESH- INDEPENDENCY FILTER//////
   dcn=zeros(nely,nelx)
   for i=1:nelx
      for j=1:nely
         summ=0.0
         for k=max(i-round(rmin),1):min(i+round(rmin),nelx)
            for l=max(j-round(rmin),1):min(j+round(rmin),nely)
               fac=rmin-sqrt((i-k)^2+(j-l)^2)
               summ=summ+max(0,fac)
               dcn(j,i)=dcn(j,i)+max(0,fac)*x(l,k)*dc(l,k)
            end
           end
           dcn(j,i)=dcn(j,i)/(x(j,i)*summ)
       end
   end
   dc=dcn

//DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
//////OPTIMALITY CRITERIA UPDATE//////
   l1=0
   l2=100000
   movee=0.2
   while(l2-l1>1e-4)
      lmid=0.5*(l2+l1)
      xnew=max(0.001,max(x-movee,min(1.,min(x+movee,x.*sqrt(-dc./lmid)))))
      for pax=1:nelx
         for pay=1:nely
            if passive(pay,pax)==1
               xnew(pay,pax)=0.001
            end
         end
      end
      if sum(sum(xnew))-volfrac*nelx*nely>0
         l1=lmid
      else
         l2=lmid
      end
   end
   x=xnew

////// FE-ANALYSIS //////
   K=sparse([],[],[2*(nelx+1)*(nely+1),2*(nelx+1)*(nely+1)])
   U=sparse([],[],[2*(nely+1)*(nelx+1),1])
   for ely=1:nely
      for elx=1:nelx
         n1=(nely+1)*(elx-1)+ely
         n2=(nely+1)*elx+ely
         edof=[2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2]
         K(edof,edof)=K(edof,edof)+x(ely,elx)^penal*KE
      end
```

```
  end

//SOLVING
  U(freedofs,:)=umfpack(K(freedofs,freedofs),"\",full(F(freedofs,:)))
  U(fixeddofs,:)=0

//OBJECTIVE FUNCTION AND SENSITIVITY
  c=0
  for ely=1:nely
    for elx=1:nelx
      n1=(nely+1)*(elx-1)+ely
      n2=(nely+1)*elx+ely
      Ue=U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1;2*n2+2; 2*n1+1;2*n1+2], 1)
      c=c+x(ely,elx)^penal*Ue'*KE*Ue
      dc(ely,elx)=-penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue
    end
  end

//PRINT RESULTS
  change=max(max(abs(x-xold)))
   disp([' it.: ' sprintf('%-4i',loop) ' obj.: ' sprintf('%-10.4f',c) 'vol.: ' sprintf('%-6.3f',sum(sum(x))/(nelx*nely))
'ch.: ' sprintf('%-6.3f',change ) 'time: ' sprintf('%-.4f', toc())])

//PLOT OBJECTIVE
  objective(loop)=c
  iteration(loop)=loop
  plot(iteration,objective)

//PLOT DENSITIES
  for elx=1:nelx
    for ely=1:nely
      n1=(elx-1)*scale
      n2=(ely-1)*scale
      xshow(1+n2:1+n2+scale,1+n1:1+n1+scale)=1-x(ely,elx)
    end
  end
  imshow(xshow)//SIVP
end.
```

7. REFERENCES

M. P. Bendsøe and O. Sigmund. Topology Optimization: Theory, Methods and Applications, Springer-Verlag, Berlin, 2003.

M. P. Bendsøe, Optimal shape design as a material distribution problem, Structural and Multidisciplinary Optimization, 1, 193-202, 1989.

M. P. Bendsøe. Optimization of Structural Topology, shape and material . Springer-Verlag, Berlin, 1995.

O. Sigmund. A 99 line topology optimization code written in Matlab. Structural and Multidisciplinary Optimization, 21, 120-127, 2001.

Scilab Documentation. https://www.scilab.org/layout/set/print/resources/documentation

Scilab Image and Video Processing Toolbox. https://atoms.scilab.org/toolboxes/SIVP

8. RESPONSIBILITY NOTICE